I Claim:

1. A computer apparatus suitable for use in the fast compilation of preverified platform neutral bytecode instructions resulting in high quality native machine code, comprising:

a central processing unit (CPU);

a computer memory coupled to said CPU, said computer memory comprised of a computer readable medium;

a compilation program embodied on said computer readable medium, said compilation program comprising:

a first code segment that receives a class file listing;

a second code segment that compiles said class file listing into machine code; and

a third code segment that interprets and executes said machine code.

2. A computer apparatus suitable for use in the fast compilation of preverified platform neutral bytecode instructions resulting in high quality native machine code, comprising:

a development or target computer system, said development or target computer system comprised of a computer readable storage medium containing a compilation program and one or more class files, said one or more class files containing one or more methods containing bytecode instruction listings;

said compilation program contained on said storage medium comprised of a first plurality of instructions, said first- plurality of instructions executed sequentially for each bytecode instruction, said first plurality configured to select first class to compile;

said compilation program contained on said storage medium comprised of a second plurality of instructions, said second plurality configured to select first method of said first class to compile;

said compilation program contained on said storage medium comprised of a third plurality of instructions, said third plurality configured to create map storage to store actual mappings and native code addresses and initialize stack mappings to empty and addresses to unknown;

said compilation program contained on said storage medium comprised of a fourth plurality of instructions, said fourth plurality configured to sequentially select each bytecode instruction in each said method of each said class file;

said compilation program contained on said storage medium comprised of a fifth plurality of instructions, said fifth plurality configured to detect stored stack maps for said selected bytecode instruction;

said compilation program contained on said storage medium comprised of a sixth plurality of instructions, said sixth plurality configured to detect direct control flow from a bytecode instruction previous to said selected bytecode instruction, said detection of direct control flow from said previous bytecode instruction resulting in said sixth plurality of instructions storing all stacks and setting said stack mappings to stack, lack of said detection of said direct control flow from said previous bytecode instruction resulting in said sixth plurality of instructions reading stack layout from said stack mappings and setting said mappings to stack;

said compilation program contained on said storage medium comprised of a seventh plurality of instructions, said seventh plurality configured to set said native code address for actual instruction;

said compilation program contained on said storage medium comprised of an eighth plurality of instructions, said eighth plurality configured to detect if said actual instruction is a load constant instruction, said detection of load constant instruction resulting in said eighth plurality of instructions creating new constant stack mapping;

said compilation program contained on said storage medium comprised of a ninth plurality of instructions, said ninth plurality configured to detect if said actual instruction is a load local instruction, said detection of load local instruction resulting in said ninth plurality of instructions creating new local stack mapping;

said compilation program contained on said storage medium comprised of a tenth plurality of instructions, said tenth plurality configured to detect if said actual instruction is a stack manipulating instruction, said detection of stack manipulating instruction resulting in said tenth plurality of instructions duplicating or reordering said stack mapping according to said stack manipulating instruction;

said compilation program contained on said storage medium comprised of an eleventh plurality of instructions, said eleventh plurality configured to detect if said actual instruction is a jump or switch instruction, said detection of jump or switch instruction resulting in said eleventh plurality of instructions emitting code using said stack mapping information and storing all said stack values not used;

said compilation program contained on said storage medium comprised of a twelfth plurality of instructions, said twelfth plurality configured to detect if said actual instruction is a remaining type of instruction, said detection of remaining type of instruction resulting in said twelfth plurality of instructions emitting code using said stack mapping information;

said compilation program contained on said storage medium comprised of a thirteenth plurality of instructions, said thirteenth plurality configured to select next instruction;

said compilation program contained on said storage medium comprised of a fourteenth plurality of instructions, said fourteenth plurality configured to select next method; and

said compilation program contained on said storage medium comprised of a fifteenth plurality of instructions, said fifteenth plurality configured to select next class file.

3. A computer implemented method for compilation of preverified platform neutral bytecode instructions resulting in high quality native machine code, comprising the steps of:

receiving a class file onto a computer readable medium containing compilation procedure instructions, said class file containing one or more methods containing platform neutral bytecode listings;

executing said compilation procedure instructions on said bytecode listings, said compilation procedure instructions sequentially processing each byte code instruction of said bytecode listing;

using information from preceding instructions to mimic an optimizing compiler; and producing native machine code on said computer readable medium.

4. A computer implemented method as recited in Claim 3 wherein said compilation procedure selects first class to compile.

5. A computer implemented method as recited in Claim 4 wherein said compilation procedure selects first method of said first class to compile.

6. A computer implemented method as recited in Claim 5 wherein said compilation procedure creates map storage to store actual mappings and native code addresses and initializes stack mappings to empty and addresses to unknown.

7. A computer implemented method as recited in Claim 6 wherein said compilation procedure sequentially selects each bytecode instruction in each said method of each said class file.

8. A computer implemented method as recited in Claim 7 wherein said compilation procedure detects stack maps for said selected bytecode instruction.

9. A computer implemented method as recited in Claim 8 wherein said compilation procedure detects direct control flow from a bytecode instruction previous to said selected actual bytecode instruction, said detection of direct control flow from said previous bytecode instruction resulting in storing all stacks and setting said stack mappings to stack, lack of said detection of said direct control flow from said previous bytecode instruction resulting in reading stack layout from said mappings and setting said mappings to stack.

10. A computer implemented method as recited in Claim 9 wherein said compilation procedure sets said native code address for actual instruction.

11. A computer implemented method as recited in Claim 10 wherein said compilation procedure detects if said actual instruction is a load constant instruction, said detection of load constant instruction resulting in said method creating new constant stack mapping.

12. A computer implemented method as recited in Claim 11 wherein said compilation procedure detects if said actual instruction is a load local instruction, said detection of load local instruction resulting in said method creating new local stack mapping.

13. A computer implemented method as recited in Claim 12 wherein said compilation procedure detects if said actual instruction is a stack manipulating instruction, said detection of stack manipulating instruction resulting in said method duplicating or reordering said stack mapping according to said stack manipulation instruction.

14. A computer implemented method as recited in Claim 13 wherein said compilation procedure detects if said actual instruction is a jump or switch instruction, said detection of jump or switch instruction resulting in said method emitting native machine code using said stack mapping information and storing all said stack values not used.

15. A computer implemented method as recited in Claim 14 wherein said compilation procedure detects if said actual instruction is a remaining type of instruction, said detection of remaining type of instruction resulting in said method emitting native machine code using said stack mapping information.

16. A computer implemented method as recited in Claim 15 wherein said compilation procedure selects next instruction.

17. A computer implemented method as recited in Claim 16 wherein said compilation procedure selects next method.

18. A computer implemented method as recited in Claim 17 wherein said compilation procedure selects next class file.

19. A computer implemented method as recited in Claim 18 wherein said compilation procedure produces said native machine code.